UNITED STATES DISTRICT COURT
DISTRICT OF MASSACHUSETTS

DATATERN, INC.,	)	
Plaintiff,	)	
v.	)	**Civil Action No. 11-11970-FDS**
MIRCROSTRATEGY, INC., et al.,	)	
Defendants.	)	

MEMORANDUM AND ORDER ON CLAIM CONSTRUCTION

SAYLOR, J.

This is a patent dispute concerning U.S. Patent No. 6,101,502, which claims a method for interfacing between two popular computerized data-storage systems: object-oriented software applications and relational databases. Plaintiff DataTern, Inc. asserts a claim of patent infringement against defendant MicroStrategy, Inc. and a number of its customers. The parties have agreed that the liability of the customer defendants turns on the liability of MicroStrategy.

The litigation is now at the claim construction stage. The parties agree on the construction of one term, "object oriented software application," and dispute the construction of six terms: (1) "class," (2) "object," (3) "object model," (4) "interface object," (5) "runtime engine," and (6) "a map of at least some relationships between schema in the database and the selected object model."

I. Background

A. Factual Background

DataTern is the owner of U.S. Patent No. 6,101,502 (the "'502 patent"). The inventors of the ’502 patent filed for the patent on September 25, 1998, claiming priority to provisional application number 60/069,157 filed on December 9, 1997. On November 10, 1999, the United States Patent and Trademark Office (“PTO”) rejected the patent’s claims as having been anticipated by Patent No. 5,627,979 (*Chang et al*.). On February 9, 2000, the inventors successfully distinguished *Chang* by amending Claims 1 and 10, which led the PTO to issue the patent on August 8, 2000.[1]

The invention claimed in the ’502 patent facilitates interaction between two popular systems for organizing computerized data: object-oriented software applications and relational databases. (Cohen Dec. ¶ 13). Object-oriented software applications encapsulate information in a collection of discrete “objects” that correspond to “classes,” which define the type of object. (*Id.* ¶ 20). For example, an object-oriented software application for a human-resources department might contain the class “employee,” which corresponds to objects representing particular employees such as “Jane Brown.” The object might contain attributes concerning Jane’s employment, such as her wage rate and scheduled hours. Relational databases organize information into rows and columns, with each column representing an attribute and each row representing an instance of those attributes. (*Id.* ¶ 29). To use the same example, the database would display a table with columns containing information about employees’ wages and hours, and rows representing a particular employee, such as Jane. The so-called “object-relational mismatch” arises because of different assumptions and approaches underlying the two systems. (*Id.* ¶ 35).

[1] In 2007, the PTO reexamined the ’502 patent following a request by a third party for *inter partes* reexamination. Initially, the PTO rejected the claims as being anticipated or rendered obvious by *Chang*, then confirmed the patentability of the original Claims 1–18 and allowed new Claims 19–44 in 2009.

The ’502 patent addresses the mismatch by generating intermediaries to translate between the systems, making the interaction easier and more reliable. (*Id.* ¶ 42–43). Representative Claim 1 describes four distinct steps the invention employs: (1) selecting an object model, (2) generating a map, (3) using the map to create at least one interface object, and (4) accessing data from the relational database via a runtime engine. ’502 patent col. 7–8 ll. 51–3.

Defendant MicroStrategy manufactures and sells products that DataTern contends infringe on the ’502 patent. DataTern contends that the infringement extends to MicroStrategy’s customers, at least some of whom are consolidated defendants in this case.

B. Procedural Background

Between November 7 and December 14, 2011, DataTern brought ten actions in this court relating to infringement of the ’502 patent. Those actions were consolidated on January 4, 2013.

Meanwhile, the ’502 patent was also the subject of litigation in the United States District Court for the Southern District of New York. That litigation involved DataTern, but not MicroStrategy. On August 24, 2012, the New York court issued a memorandum and order on claim construction concerning two consolidated cases, *Microsoft Corp. v. DataTern, Inc.*, No. 11-cv-02365-KBF (S.D.N.Y. filed Apr. 7, 2011, and *SAP et al v. DataTern, Inc.*, No. 11-cv-02648-KBF (S.D.N.Y. filed Apr. 18, 2011). *Microsoft Corp. v. DataTern, Inc.*, 2012 WL 3682915, at *10 (S.D.N.Y. Aug. 24, 2012) (“*Microsoft I*”). Among other things, the court construed certain terms in the ’502 patent, including the terms “to create at least one interface object” and “object model.” Thereafter, DataTern conceded noninfringement based on the court’s construction of the claim terms and the court entered summary judgment. DataTern appealed.

On February 7, 2013, based on the concession by DataTern that MicroStrategy could not be held to have infringed the ’502 patent based on the *Microsoft I* court’s construction of “create at least one interface object,” this Court granted summary judgment to MicroStrategy. (Docket No. 108). DataTern also appealed that judgment.

On May 5, 2014, the Federal Circuit upheld the *Microsoft I* court’s judgment of non-infringement. *Microsoft Corp. v. DataTern, Inc.*, 755 F.3d 899, 909 (Fed. Cir. 2014) (*“Microsoft II*”). In so doing, it upheld the New York district court’s determination that the term “object model” included classes. *Id.* It did not, however, review the district court’s construction of “to create at least one interface object,” as that proved unnecessary to render its decision. *Id.* at 908–909.

On December 19, 2014, the Federal Circuit vacated this Court’s order of summary judgment, finding that the New York court’s construction of the term “to create at least one interface object” (which this Court had essentially adopted) was incorrect. *DataTern, Inc. v. Epicor Software Corp.*, 599 F. App'x 948, 954-55 (Fed. Cir. 2014). It construed the term “to create at least one interface object” to mean “to instantiate at least one interface object from a class.” *Id*. The Federal Circuit remanded the case to this Court for further proceedings. *Id.* at 955.

On February 11, 2015, MicroStrategy filed two separate motions for summary judgment, one on the basis of invalidity for non-patentable subject matter and the other on the basis of non-infringement. On September 4, 2015, the Court denied both motions.

The action is now at the claim construction stage. On September 26, 2016, the Court held a *Markman* hearing on the six disputed claim terms.

II. <u>Legal Standard</u>

The construction of claim terms is a question of law. *Markman v. Westview Instruments,* 517 U.S. 370, 372 (1996) ("[T]he construction of a patent, including terms of art within its claim, is exclusively within the province of the court.").

In *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) (*en banc*), the Federal Circuit clarified the proper approach to claim construction and set forth principles for determining the hierarchy and weight of the definitional sources that give a patent its meaning. The guiding principle of construction is "the meaning that the term would have to a person of ordinary skill in the art in question at the time of . . . the effective filing date of the patent application." *Id.* at 1313. Courts thus seek clarification of meaning in "the words of the claims themselves, the remainder of the specification, the prosecution history, and extrinsic evidence concerning relevant scientific principles, the meaning of technical terms, and the state of the art." *Id.* at 1314 (quoting *Innova/Pure Water, Inc. v. Safari Water Filtration Sys.*, 381 F.3d 1111, 1116 (Fed. Cir. 2004)).

A. <u>The Words of the Claim</u>

The claim construction analysis normally begins with the claims themselves.[2] The claims of a patent "define the invention to which the patentee is entitled the right to exclude." *Phillips*, 415 F.3d at 1312 (citing *Innova*, 381 F.3d at 1115).

A court may construe a claim term to have its plain meaning when such a construction

[2] In *Phillips*, the Federal Circuit discredited the practice of starting the claim construction analysis with broad definitions found in dictionaries and other extrinsic sources:

> [I]f the district court starts with the broad dictionary definition . . . and fails to fully appreciate how the specification implicitly limits that definition, the error will systematically cause the construction of the claim to be unduly expansive. The risk of systematic overbreadth is greatly reduced if the court instead focuses at the outset on how the patentee used the claim term in the claims, specification, and prosecution history, rather than starting with a broad definition and whittling it down.

415 F.3d at 1321. Of course, if no special meaning is apparent after reviewing the intrinsic evidence, claim construction might then "involve[] little more than the application of the widely accepted meaning of commonly understood words." *Id.* at 1314.

resolves a dispute between the parties. *See O2 Micro Int'l Ltd. v. Beyond Innovation Tech. Co.*, 521 F.3d 1351, 1361 (Fed. Cir. 2008); *see also U.S. Surgical Corp. v. Ethicon, Inc.*, 103 F.3d 1554, 1568 (Fed. Cir. 1997) ("Claim construction is a matter of resolution of disputed meanings and technical scope, to clarify and when necessary to explain what the patentee covered by the claims, . . . [but] is not an obligatory exercise in redundancy.").

In some instances, it is the arrangement of the disputed term in the claims that is dispositive. "This court's cases provide numerous . . . examples in which the use of a term within the claim provides a firm basis for construing the term." *Phillips*, 415 F.3d at 1314. For example, because claim terms are normally used consistently throughout the patent, the meaning of a term in one claim is likely the meaning of that same term in another. *Id.* In addition, "the presence of a dependent claim that adds a particular limitation gives rise to a presumption that the limitation in question is not present in the independent claim." *Id.* at 1315.

B. The Specification

"The claims, of course, do not stand alone." *Id.* at 1315. Rather, "they are part of a fully integrated written instrument, consisting principally of a specification that concludes with the claims." *Id.* (citations and quotations omitted). For that reason, the specification must always be consulted to determine a claim's intended meaning. The specification "is always highly relevant to the claim construction analysis. Usually, it is dispositive; it is the single best guide to the meaning of a disputed term." *Id.* (quoting *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996)).

"In general, the scope and outer boundary of claims is set by the patentee's description of his invention." *On Demand Mach. Corp. v. Ingram Indus.*, 442 F.3d 1331, 1338 (Fed. Cir. 2006); *see also Phillips*, 415 F.3d at 1315–17 ("[T]he interpretation to be given a term can only be determined and confirmed with a full understanding of what the inventors actually invented and intended to envelop with the claim.") (quoting *Renishaw*, 158 F.3d at 1250). "[T]he specification may reveal a special definition given to a claim term by the patentee that differs from the meaning it would otherwise possess." *Phillips*, 415 F.3d at 1316. It may also reveal "an intentional disclaimer, or disavowal, of claim scope by the inventor." *Id.* Therefore, the claims are to be construed in a way that makes them consistent with, and no broader than, the invention disclosed in the specification. *On Demand*, 442 F.3d at 1340 ("[C]laims cannot be of broader scope than the invention that is set forth in the specification."); *Phillips,* 415 F.3d at 1316 ("[C]laims must be construed so as to be consistent with the specification, of which they are a part.") (quoting *Merck & Co. v. Teva Pharms. USA, Inc.*, 347 F.3d 1367, 1371 (Fed.Cir.2003)).

Nevertheless, courts must be careful to "us[e] the specification [only] to interpret the meaning of a claim" and not to "import[] limitations from the specification into the claim." *Id.* at 1323. A patent's "claims, not specification embodiments, define the scope of patent protection." *Kara Tech. Inc. v. Stamps.com Inc.*, 582 F.3d 1341, 1348 (Fed. Cir. 2009); *see also Martek Biosciences Corp. v. Nutrinova, Inc.*, 579 F.3d 1363, 1381 (Fed. Cir. 2009) ("[E]mbodiments appearing in the written description will not be used to limit claim language that has broader effect."). "In particular, we have expressly rejected the contention that if a patent describes only a single embodiment, the claims of the patent must be construed as being limited to that embodiment." *Phillips*, 415 F.3d at 1323. This is "because persons of ordinary skill in the art rarely would confine their definitions of terms to the exact representations depicted in the embodiments." *Id.*

Although this distinction "can be a difficult one to apply in practice[,] . . . the line between construing terms and importing limitations can be discerned with reasonable certainty and predictability if the court's focus remains on understanding how a person of ordinary skill in the art would understand the claim terms." *Id.* "The construction that stays true to the claim language and most naturally aligns with the patent's description of the invention will be, in the end, the correct construction." *Id.* at 1316 (quoting *Renishaw PLC v. Marposs Societa' per Azioni*, 158 F.3d 1243, 1250 (Fed. Cir. 1998)).

C. <u>The Prosecution History</u>

After the specification and the claims themselves, the prosecution history is the next best indicator of term meaning. The prosecution history consists of the complete record of the proceedings before the PTO and includes the prior art cited during the examination of the patent. *Id.* at 1317. "Like the specification, the prosecution history provides evidence of how the PTO and the inventor understood the patent." *Id.* "[T]he prosecution history can often inform the meaning of the claim language by demonstrating how the inventor understood the invention and whether the inventor limited the invention in the course of prosecution, making the claim scope narrower than it would otherwise be." *Id.* (citing *Vitronics*, 90 F.3d at 1582–83).

However, "because the prosecution history represents an ongoing negotiation between the PTO and the applicant, rather than the final product of that negotiation, it often lacks the clarity of the specification and thus is less useful for claim construction purposes." *Id.* As a result, courts generally require that "a patent applicant . . . clearly and unambiguously express surrender of [a] subject matter" to disavow claim scope during prosecution. *Voda v. Cordis Corp.*, 536 F.3d 1311, 1321 (Fed. Cir. 2008) (quoting *Sorensen v. Int'l Trade Comm'n*, 427 F.3d 1375, 1378 (Fed. Cir. 2005)).

D. Extrinsic Sources

Extrinsic evidence consists of "all evidence external to the patent and prosecution history, including expert and inventor testimony, dictionaries, and learned treatises." *Phillips*, 415 F.3d at 1317 (quoting *Markman*, 52 F.3d at 980). It "can help educate the court regarding the field of the invention and can help the court determine what a person of ordinary skill in the art would understand claim terms to mean." *Id.* at 1319. However, extrinsic evidence suffers from a number of defects, including its independence from the patent, potential bias, and varying relevance. *Id.* at 1318–19. Such evidence is therefore "unlikely to result in a reliable interpretation of patent claim scope unless considered in the context of the intrinsic evidence," and courts may consider, or reject, such evidence at their discretion. *Id.* at 1319.

III. Analysis

The disputed terms occur throughout the '502 patent, but all appear in representative Claim 1. Claim 1 states as follows:

> **1.** A method for interfacing an *object oriented software application* with a relational database, comprising the steps of:
>
> selecting an *object model*;
>
> generating *a map of at least some relationships between schema in the database and the selected object model*;
>
> employing the map to create at least one *interface object* associated with an *object* corresponding to a *class* associated with the object oriented software application; and
>
> utilizing a *runtime engine* which invokes said at least one interface object with the object oriented application to access data from the relational database.

(emphasis added).

The parties' proposed constructions of the disputed terms are as follows:

	Plaintiff's Proposed	**Defendant's Proposed**

	Construction	**Construction**
“Object oriented software application”	software application organized as a collection of classes	software application organized as a collection of classes
“Object model”	a template with a predetermined, standardized structure that has classes	a template with a predetermined standardized structure both relating to an object-oriented software application and including object classes and inheritance relationships among classes
“A map of at least some relationships between schema in the database and the selected object model”	no construction needed	a declarative structure that describes at least one association between the objects and attributes of an object model and/or tables, rows, and/or columns in the database classes
“Interface object”	interface objects “act as intermediaries between the object oriented application and the relational database”	an object, which is not part of or generated by the object oriented software application, that provides a connection between the object oriented software application and the runtime engine, and that accesses data from the database in accordance with the map
“Object”	an entity comprising one or more attribute values (i.e., data) and/or behaviors (i.e., functionality)	an entity comprising attribute values (i.e., data) and methods (i.e., functionality) that can act on said attribute values and that accesses data from the database in accordance with the map
“Class”	a definition that specifies one or more attributes and/or behaviors of objects	a definition that specifies attributes and behavior of objects, and from which objects can be instantiated
“Runtime engine”	software that the object oriented software application uses to access the relational database	software that (i) the object oriented software application depends on to run, (ii) must be running to execute the object oriented software application, (iii) uses the map in its processing, and (iv) is not part of the object oriented software application

A. <u>“Object Oriented Software Application”</u>

	Plaintiff’s Proposed Construction	**Defendant’s Proposed Construction**

"Object oriented software application"	software application organized as a collection of classes	software application organized as a collection of classes

The parties have agreed as to the construction of "object oriented software application." Accordingly, the Court will adopt the agreed-upon construction of "software application organized as a collection of classes."

B. "Class"

	Plaintiff's Proposed Construction	**Defendant's Proposed Construction**
"Class"	a definition that specifies one or more attributes and/or behaviors of objects	a definition that specifies attributes and behavior of objects, and from which objects can be instantiated

MicroStrategy's proposed definition mirrors the definition to which DataTern stipulated in *Microsoft I.* MicroStrategy contends that DataTern should be judicially estopped from advancing a different construction from the one to which it stipulated in *Microsoft I.* In the alternative, it contends that its narrower definition is correct on the merits. The Court will address those arguments in turn.

1. Judicial Estoppel

MicroStrategy first contends that DataTern should be judicially estopped from advancing a definition of "class" that is inconsistent with the definition to which it stipulated in *Microsoft I*. 2012 WL 3682915, at *4. In that proceeding, DataTern stipulated that "class" meant "a definition that specifies attributes and behavior of objects, and from which objects can be instantiated." *Id.*

Judicial estoppel is an equitable doctrine that prevents a party from assuming contrary legal positions across judicial proceedings "simply because [its] interests have changed." *New Hampshire v. Maine*, 532 U.S. 742, 749 (2001). Although not an exhaustive test, at least three conditions must be satisfied for the doctrine to apply: (1) the earlier and later positions must be "clearly inconsistent"; (2) the party to be estopped "must have succeeded in persuading a court to accept the earlier position"; and (3) the party to be estopped "must stand to derive an unfair advantage if the new position is accepted by the court." *RFF Family P'ship, LP v. Ross*, 814 F.3d 520, 528 (1st Cir. 2016) (quotations omitted). Here, MicroStrategy has demonstrated that the first two elements have been satisfied. It has not, however, demonstrated that DataTern would stand to derive an unfair advantage if the Court adopts the new construction.

First, the constructions advanced by DataTern in the prior and current litigation are "clearly inconsistent." DataTern attempts to reconcile that facial inconsistency by stating that it originally stipulated to the meaning of "class" only "as the term appears in the claims." Pl. Rebuttal Brief at 14; *see also id.*, Ex. M at 2 (stating that the "agreed construction" applied to claims 1, 2, 7, 10, 11, and 16). Both the *Microsoft I* and *Microsoft II* courts found that the term "object model" necessarily includes classes. *See Microsoft I*, 2012 WL 3682915, at *6, *Microsoft II*, 755 F.3d at 909. DataTern contends that the stipulated definition does not apply to those courts' importation of the definition of "class" to the definition of "object model." However, the *Microsoft I* court adopted the disputed definition of "object model" as comprising "classes" concurrently with accepting the stipulated definition of "class." *Microsoft I*, 2012 WL 3682915, at *6. Similarly, the *Microsoft II* court approved the *Microsoft I* court's construction of "object model" as comprising "classes" and acknowledged that DataTern was "bound" by its stipulated construction. *Microsoft II*, 755 F.3d at 909. Therefore, both courts construed "object model" to comprise "classes" with the understanding that a "class" was defined according to DataTern's stipulated definition. The courts did not limit the application of the "class" definition to certain terms. Here, too, DataTern cannot circumscribe this definition only to the circumstances that benefit it.

Second, it is plain that DataTern persuaded the earlier court to accept its position. In *Microsoft I*, DataTern advanced a particular definition, and the court adopted that definition. Thus, DataTern "succeeded in persuading a court to accept the earlier position." *RFF Family P'ship, LP*, 814 F.3d at 528. The fact that DataTern lost in that proceeding on other grounds is not relevant to whether it succeeded on this issue. DataTern contends that the *Microsoft I* court rejected its definition because it stated that the "intrinsic evidence suggests that classes need not always include behaviors." *Microsoft I*, 2012 WL 3682915, at *6. It offers further support for that proposition by pointing to language in the *Microsoft II* decision in which the Federal Circuit stated that the stipulated definition is "a narrower construction of classes than that required by the '502 patent." *Microsoft II*, 755 F.3d at 909. Although DataTern characterizes these statements as holdings, they are not essential to the judgment in either case, and are better labeled dicta adopted without the benefit of full briefing. The issue was not before those courts, because DataTern not only did not challenge SAP's definition of "class," but also stipulated to a definition that included behaviors. The *Microsoft I* court accepted that stipulated definition. *See Microsoft I*, 2012 WL 3682915, at *4. Dicta expressing doubt about the wisdom of the stipulation cannot alter that fact. Accordingly, DataTern succeeded in persuading the earlier court to accept its position.

Despite satisfying the first two prongs of the judicial estoppel test, MicroStrategy has not demonstrated that DataTern stands to derive an "unfair advantage" if its new definition is accepted by this Court. First, it is relevant that DataTern did not espouse its original position adversely to MicroStrategy. *See Davis v. Wakelee*, 156 U.S. 680, 689–90 (1895) (finding that a party may not advance inconsistent positions in different legal proceedings "especially if it be to the prejudice of the party who has acquiesced in the position formerly taken by him.”). The earlier proceeding was brought against a third party, SAP. SAP is not a party to this lawsuit and will experience no obvious prejudice as a result of DataTern’s changed position. Second, although DataTern received the benefit the court’s acceptance of its stipulated definition, it did not receive that benefit to the loss of its adversary. Instead, both DataTern and SAP presumably benefitted from the court’s construction, because they agreed to the construction. Because of the non-adversarial posture of the issue, the stipulated definition was not closely examined by either the *Microsoft I* or the *Microsoft II* courts, and both courts, as DataTern points out, expressed some doubt about whether it was correct. Here, finding a definition of “class” that is inconsistent with the parties’ unexamined, stipulated definition does not undermine the “integrity of the judicial process” in the same way or to the same extent that finding an inconsistent definition would were the original definition adopted after an adversarial process. *New Hampshire v. Maine*, 532 U.S. 742, 749 (2001).

Under these circumstances, it would not unfairly prejudice MicroStrategy to decide this question on the merits, nor would it extend an unfair advantage to DataTern. Accordingly, the Court does not find that DataTern is judicially estopped from advancing a definition inconsistent with the one to which it stipulated in *Microsoft I*.

2. **Merits**

Turning to the merits, the parties dispute the necessity of including both “attributes” and “behaviors” in the construction of “class,” and the inclusion of the phrase “from which objects may be instantiated.”

a. **Behavior and Instantiation**

As it has been argued, there is no practical difference between the necessity of “behavior” and the inclusion of the phrase "from which objects can be instantiated." MicroStrategy contends that the term "class" must specify "behavior," because all classes must have at least one method: the constructor method. That is, all classes must be able to construct, or instantiate, objects, and therefore behavior is inherent in the definition of class.

Some basic background on the term "class" in the context of object-oriented programming is required. As described above, object-oriented programs organize data into discrete units, called "objects." *See* Cohen Dec. ¶ 20. Individual objects are defined by the "classes" to which they belong. *See id.* ¶ 22. A given class, such as "employee," specifies the attributes and behaviors associated with the class, such as "name" and "assign a new project." *See id*. The term "class" has been variously analogized to a blueprint, a factory, or a cookie cutter. *See id.* ¶ 24; *Microsoft I*, 2012 WL 3682915, at *4. Just as a blueprint guides the construction of buildings, a factory makes goods, and a cookie cutter cuts cookies, the ultimate purpose of a class is to instantiate objects. Toward this end, classes also permit "inheritance" relationships in which a sub-class inherits all the traits of another class and specifies new attributes or behaviors. *See* Cohen Dec. ¶ 23. For example, "hourly worker" and "salaried worker" classes might inherit attributes and behaviors from the "employee" class, but add traits specific to those sub-classes. *Id.*

The term "class" appears repeatedly throughout the claims in the '502 patent. Most importantly, independent claims 1 and 10 refer to the term. Those claims state:

1. A method for interfacing an object oriented software application with a relational database, comprising the steps of:

   selecting an object model;

   generating a map of at least some relationships between schema in the database and the selected object model;

   employing the map to **create at least one interface object associated with an object corresponding to a class** associated with the object oriented software application; and

utilizing a runtime engine which invokes said at least one interface object with the object oriented application to access data from the relational database;

10. A computer program fixed on a computer-readable medium and adapted to operate on a computer to provide access to a relational database for an object oriented software application, comprising:

a mapping routine that generates a map of at least some relationships between schema in the database and a selected object model;

a code generator that employs said map **to create at least one interface object associated with an object corresponding to a class** associated with the object oriented software application

a runtime engine that invokes said at least one interface object to access data from the relational database.

’502 patent col. 7-8 ll. 53–23 (emphasis added). As described in the claims of the ’502 patent, a “class” corresponds to an “object,” which in turn is associated with “interface objects.” The ’502 patent repeatedly refers to objects as “instances” of classes. *See* ’502 patent col. 2 l. 55; *id*. col. 4 l. 12, 32–33, 39–40, 58–59, 60–61; *id*. col. 6 ll. 26–27; *id*. col. 7 ll. 14, 28. The parties’ experts agree that objects are “instances” of classes. *See* Cohen Dec. ¶22; McGoveran Dec. ¶ 25. In the context of claims 1 and 10, a class is not the inert entity that DataTern posits.[3] Instead, the plain meaning of these claims is that classes correspond with objects by instantiating them. It follows that the classes claimed in the ’502 patent must include behavior because they instantiate objects.

[3] At oral argument and briefly in a footnote in its third and final submission on claim construction, DataTern drew attention to the patent’s reference to so-called “abstract classes” in the detailed description of the invention, as support for the proposition that classes need not instantiate objects. Pl. Post-Hearing Brief at 5 n.3; ’502 patent col. 6 ll. 9–11. That argument was not raised in DataTern’s initial or responsive claim construction briefs. Raising the argument at this late stage did not give MicroStrategy fair opportunity to respond, because the argument was submitted in written form concurrently with MicroStrategy’s final brief. Accordingly, the Court finds that DataTern has waived the argument. In any event, although the Court does not have the benefit of full briefing, it appears incorrect on the merits. It is true that abstract classes do not directly instantiate objects. McGoveran Dec. ¶ 30. Instead, they form the basis of inheritance relationships that are passed on to generated “concrete” classes, which in turn instantiate objects. *Id.* The abstract class referred to in Figures 5 and 6, OOBase, is capable of instantiating objects indirectly by passing on information to OObject, which in turn instantiates objects. *Id.* ll. 9-30; fig. 5–6. OOBase also clearly contains a method, “GetAttrPtr.” *Id.* fig. 6. Although this reference to abstract classes suggests that not all classes need instantiate objects directly, it does not undermine the finding that the “classes” described in the patent claims, which “correspond” to objects, instantiate those objects.

DataTern further contends that other intrinsic evidence supports a finding that objects need not have behaviors. DataTern points to Figure 3 of the patent, which depicts "a block diagram of an object model." '502 patent col. 2 l. 18; *id.* fig. 3. Figure 3 represents four classes—CPerson, CDepartment, CProject, and CEmployee—which include a list of attributes, but do not include behaviors:

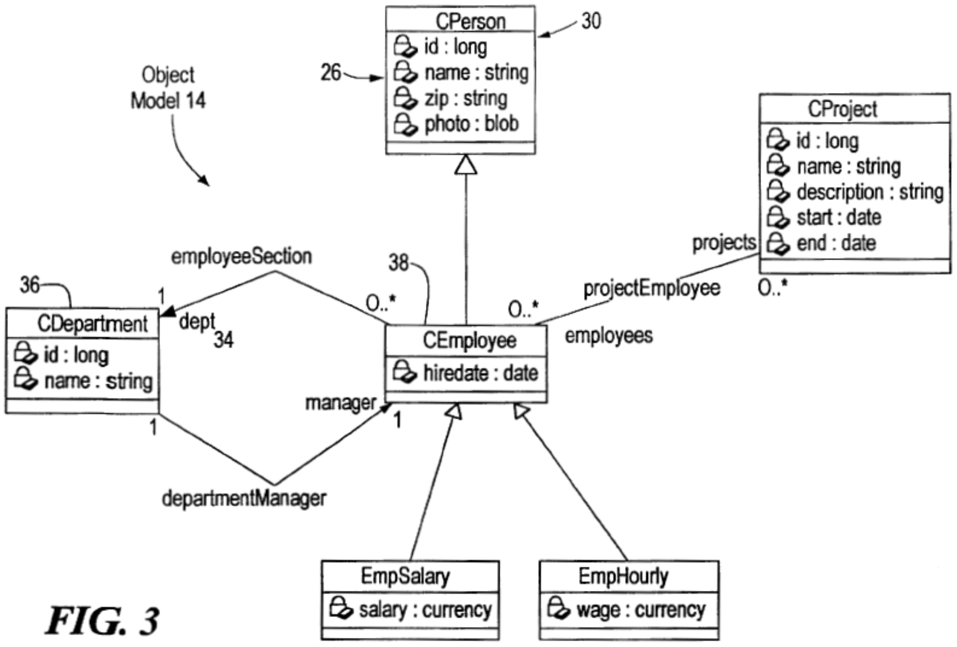


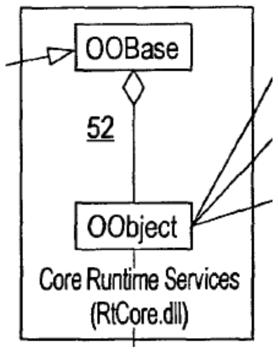
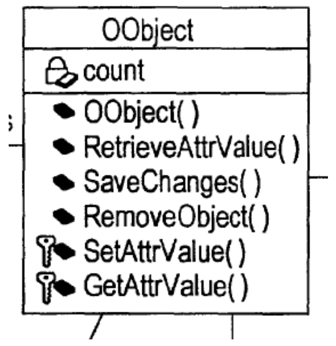
FIG. 3

Id. fig. 3. DataTern contends that based on this diagram, the intrinsic evidence supports a finding that "class" need not include behaviors, because the classes depicted do not include them. As further support, DataTern cites language from *Microsoft II* that interprets this diagram to disclose an "object model with classes and attributes of those classes," but not mentioning behaviors. *Microsoft II,* 755 F.3d at 909. DataTern also points to language from *Microsoft I* in which the court interpreted the diagram, concluding that "[t]he intrinsic evidence suggests that classes need not always include behaviors." *Microsoft I,* 2012 WL 3682915, at *6. Again, those statements were made without the benefit of full briefing on this issue, because DataTern stipulated to a definition of class that included MicroStrategy's proposed limitations.

The court can only use a specification "to interpret the meaning of a claim." *Phillips*, 415

F. 3d at 1323. The patent's "claims, not [its] specification embodiments, define the scope of patent protection." *Kara Tech.*, 582 F.3d at 1348. Figure 3 does not purport to be an exhaustive depiction of a class. Indeed, the purpose of the diagram is not to represent a "class" at all, but instead, to represent a "block diagram of an object model." '502 patent col. 2 l. 18. In other areas of the patent, block diagrams are explicitly underinclusive. For example, Figure 5 represents the class OObject, but does not depict any behaviors associated with this class. '502 patent fig. 5. In contrast, Figure 6 provides a more detailed depiction of OObject, which includes behaviors:

Figure 5 Depiction of OObject	**Figure 6 Depiction of OObject**

 OOBase
52
OObject
Core Runtime Services
(RtCore.dll)
	 OObject
	count
	OObject( )
	RetrieveAttrValue( )
	SaveChanges( )
	RemoveObject( )
	SetAttrValue( )
	GetAttrValue( )

'502 patent fig. 5–6. That discrepancy is strong evidence that the block diagrams contained in the patent are, in at least some respects, non–exhaustive. The Court does not find that the omission of behaviors in the classes depicted in Figure 3 contradicts the language of the patent's claim, which includes repeated and consistent reference to their corresponding objects. In the context of the '502 patent claims, classes instantiate objects, so it follows that the term comprises behavior.

b. <u>Attributes</u>

Again, whether a class must contain "attributes," or simply may contain them, has to be determined within the context of the purpose and claims of the '502 patent. The purpose of the

’502 patent is to “interfac[e] object oriented software applications with relational databases.” ’502 patent col. 1 ll. 23–24. Again and again, the patent instructs that it facilitates interfacing between the class or object attributes and values in a relational database.[4] DataTern’s own expert states that the “’502 patent is concerned with mapping attributes between an object model and a database schema . . . .” Cohen Dec. ¶ 15. If a class did not contain attributes, there would be nothing to map. The only definition of “class” consistent with the patent’s claims and purpose is one that requires the presence of attributes.[5]

3. Conclusion

For the foregoing reasons, the term “class” will be construed to mean “a definition that specifies attributes and behavior of objects, and from which objects can be instantiated.”

C. Object

	Plaintiff’s Proposed Construction	**Defendant’s Proposed Construction**
“Object”	an entity comprising one or more attribute values (i.e., data) and/or behaviors (i.e., functionality).	an entity comprising attribute values (i.e., data) and methods (i.e., functionality) that can act on said attribute values

As it did for its proposed construction of “class,” MicroStrategy advances a construction of the term “object” that is consistent with the definition to which DataTern stipulated in

[4] For example, the patent states as follows: “Mapping the object model to the relational database schema includes mapping a class attribute to a table column, mapping a class attribute to a 1-1, 1-N, or N-N relationship,” ’502 patent col. 2 ll. 44–47; “mapping of a class attribute to a table column can be described,.” *id.* col. 2 ll. 49–50; “the class attribute CPerson.name 26 maps to table column TPerson.name,” *id.* col. 2 ll. 53-54; and “Schema relationships are mapped directly to object relationships, either in the form of object attributes or list attributes,” *id.* col. 5 ll. 29–30. Claim 32 discloses “the method of Claim 1 wherein said utilizing is further defined as the object oriented software application invoking at least one interface object to request data from the relational database corresponding to at least one attribute of at least one object corresponding to a class associated with the object oriented software application.” *Id.* col. 2 ll. 10–12.

[5] Again, DataTern argues in a footnote in its post-hearing brief and at oral argument that the presence of a “stateless” object in Claim 9 suggests that attributes are a potential but not necessary feature of classes. Pl. Post-Hearing Brief at 5 n.2. Again, the Court finds that it has waived that argument by failing to advance it earlier in the briefing.

Microsoft I. Again, MicroStrategy contends that DataTern should be judicially estopped from advancing a contrary definition here. For the reasons described above, DataTern is not judicially estopped from advancing an inconsistent definition, and the Court will reach the merits of the dispute.

On the merits, the parties dispute the whether the term "object" must comprise methods and attributes, and, if so, whether the methods must be able to act on the attributes.

1. **<u>Necessity of Methods and Attributes</u>**

As described, a "class" must contain attributes and behaviors. That analysis can be applied with equal force to the construction of the term "object," which is simply an instance of a class.[6] The parties agree that their arguments concerning the necessity of methods and attributes in the definition of "object" rises and falls with their arguments concerning the necessity of behaviors in the definition of "class." *See* Pl. Claim Construction Brief at 24–25; Pl. Responsive Brief at 4–5; Def. Claim Construction Brief at 19. Accordingly, the Court finds that an "object" must comprise methods and attribute values.

2. **<u>Methods Can Act on Attributes</u>**

According to MicroStrategy, the concept of "encapsulation" requires that an object's methods can act on its attributes. The patent itself does not make reference to encapsulation, but the experts for both sides refer to the concept. *See* Cohen Dec. ¶ 20; McGoveran Dec. ¶¶ 22, 24, 27, 42, 58, 63. McGoveran identifies encapsulation as a "defining principle" of object-oriented programming that is the source of "[m]ost of the object relational impedance mismatch." McGoveran Dec. ¶¶ 22, 63. He states that an encapsulated object "prevents other objects from accessing [all the data pertaining to it]." *Id.* ¶¶ 58. Similarly, Cohen states that data is

[6] The parties agree that "behavior" in the definition of "class" is the same as "methods" in the proposed definition of "object." Cohen Dec. ¶ 20; McGoveran Dec. ¶ 25.

encapsulated in objects which are, in turn, "insulate[d]" from one another. Cohen Dec. ¶ 21.

MicroStrategy contends that because objects are encapsulated, the only way that attributes can be changed or otherwise acted upon is through the methods of the object. *See* Pl. Claim Construction Brief at 20. DataTern does not specifically refute that claim, instead relying on the contention that because an object need not specify attributes or behavior, it follows that a method need not act on an attribute. In fact, DataTern's expert states that "[e]ach object encapsulates a set of data and, often, a set of behaviors (sometimes known as "methods") that can operate on that data." Cohen Dec. ¶ 20. Although Cohen does not concede that an object must contain methods, this statement suggests that if an object does contain methods, those methods act on the object's attributes. Having found that an "object" within the context of the '502 patent must contain methods, it now follows that those methods are capable of acting on the object's attributes.

3. Conclusion

For the foregoing reasons, the term "object" will be construed to mean "an entity comprising attribute values (i.e., data) and methods (i.e., functionality) that can act on said attribute values."

D. Object Model

	Plaintiff's Proposed Construction	**Defendant's Proposed Construction**
"Object model"	a template with a predetermined, standardized structure that has classes	a template with a predetermined standardized structure both relating to an object-oriented software application and including object classes and inheritance relationships among classes

As with the terms "class" and "object," MicroStrategy advances a definition that is consistent with the definition found in *Microsoft I,* while DataTern advances a different definition. However, unlike for the previously discussed terms, DataTern contested the construction of "object model" in *Microsoft I* and *Microsoft II*, and those courts construed the term adverse to DataTern. Only the Federal Circuit's decision in *Microsoft II* is controlling here.

In *Microsoft I,* the district court found that an "object model" is "a template with a predetermined standardized structure both relating to an object-oriented software application and including object classes and inheritance relationships among classes." *Microsoft I,* 2012 WL 3682915, at *6. DataTern appealed that finding, and the Federal Circuit affirmed the decision insofar as the court required the term to include "classes." *See Microsoft II,* 755 F.3d at 908 n.6 ("DataTern also challenges the district court's determination that object model requires inheritance relationships among classes and that the object model be related to the object-oriented software application. Because the requirement of classes is dispositive, we do not address the other aspects of the court's claim construction of object model.")

Although this court is not bound by the limitations placed on the term by the *Microsoft I* court, the reasoning in *Microsoft I* is obviously instructive; indeed, precisely the same issue is before the court. *See, e.g., GeoTag, Inc. v. Fred's, Inc.,* 2013 WL 2181166, at *6 (W.D. Tenn. May 20, 2013) (finding that another district court's claim construction is not binding, and that "it may be appropriate to give deference to the previous court's construction if the same terms are relitigated"). Here, as there, there are two issues in dispute.[7] First, the parties dispute whether an "object model" is required to "relat[e] to an object-oriented software application." Second, the parties dispute whether the "object model" must include "inheritance relationships among classes."

[7] The Court interprets the phrases "that has classes" and "including object classes" in the parties' proposed definitions to have substantially the same meaning.

1. **Relationship to the Object-Oriented Software Application**

The term "object model" appears in Claim 1 in the phrases "selecting an object model" and "generating a map of at least some relationships between schema in the database and the selected object model." '502 patent col. 7–8 ll. 53–4. The term also appears in Claim 10 in the phrase "a mapping routine that generates a map of at least some relationships between schema in the database and a selected object model." *Id.* col. 8 ll. 24–37.

MicroStrategy contends that an "object model" must relate to an object-oriented software application in order to reflect the scope of that term as it appears in the '502 patent. *See* Def. Claim Construction Brief at 28. It argues that without that limitation, the term "object model" would be so broad as to encompass even Microsoft Excel spreadsheets. *Id.* DataTern responds that MicroStrategy's construction is not required by the terms of the patent. *See* Pl. Claim Construction Brief at 15.

In *Microsoft I,* the court found that "the context of the patent is critical" in determining whether an object model must relate to an object-oriented software application. *Microsoft*, 2012 WL 3682915, at *6. The court pointed to language in the abstract and summary of the invention demonstrating that "the invention requires an object-oriented software application," so "it is appropriate to include that language in the construction of 'object model.'" *Id.*; '502 patent at [57] ("The Database schema, object model and mapping are employed to provide interface objects that are utilized by a runtime engine to facilitate access to the relational database by object oriented software applications."); '502 patent col. 1 ll. 59–62. Based on that analysis of the language and purpose of the '502 patent, the court concluded that, in context, the object model was limited "to one involving object-oriented software applications." *Microsoft I*, 2012 WL 3682915, at *6.

The Court finds the *Microsoft I* court's interpretation of the term "object model" persuasive. Accordingly, the term "object model" as used in the patent must relate to an object-oriented software application.

2. **Inheritance Relationships Among Classes**

The *Microsoft I* court also concluded that an "object model" must include inheritance relationships among classes. However, the court did not explain its reasoning for finding that such a restriction applied. *See Microsoft*, 2012 WL 3682915, at *5-7.

DataTern contends that although the object model may specify inheritance relationships, such relationships are not required by the patent. MicroStrategy responds that the only embodiment of an object model in the patent, Figure 3, depicts inheritance relationships among classes, and therefore object models must contain inheritance relationships. *See* '502 patent fig. 3. However, "it is improper to read limitations from a preferred embodiment described in the specification—even if it is the only embodiment—into the claims absent a clear indication in the intrinsic record that the patentee intended the claims to be so limited." *Epicor,* 599 F. App'x at 952 n.3 (quoting *Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 913 (Fed. Cir. 2004)). The intrinsic evidence to which MicroStrategy points indicates that an object model may, but need not, include inheritance relationships.

To the extent that Figure 3 creates an ambiguity, MicroStrategy has not demonstrated that the extrinsic evidence supports the conclusion that an object model must contain inheritance relationships. It has submitted extrinsic evidence suggesting that an object model must represent "relationships" among classes. *See* Def. Ex. 29, James Rumbaugh et al., *Object-Oriented Modeling and Design* (Prentice-Hall 1991) at 17; Def. Ex. 30, *FireStar Software, Inc. v. Red Hat, Inc.*, et al 2:06-cv-258, Joint Claim Construction and Prehearing Statement dated March 13, 2008 at 2 (“*FireStar v. Redhat* Jt. Claim Construction Stmt.”). DataTern, however, disputes that the term “relationships” is coincident with the term “inheritance relationships.” *See* Cohen Rebuttal Dec. ¶ 30. Indeed, MicroStrategy’s own expert refers to “relationships” between classes that are not “inheritance relationships.” *See* McGoveran Dec. ¶ 26 (stating that a class attribute may correspond to another class “creat[ing] a relationship between the class containing the attribute and the class defining the attribute type.”); *id.* ¶ 34 (stating that an object model describes “relationships among classes, especially inheritance” but not limiting those relationships to inheritance relationships). The extrinsic evidence accordingly does not require the conclusion that an object model must specifically include “inheritance relationships,” and the Court will not read that limitation into the patent claims.

3. <u>Conclusion</u>

For the foregoing reasons, the term “object model” will be construed to mean “a template with a predetermined standardized structure both relating to an object-oriented software application and including object classes.”

E. <u>Interface Object</u>

	Plaintiff’s Proposed Construction	**Defendant’s Proposed Construction**
“Interface object”	interface objects “act as intermediaries between the object oriented application and the relational database.”	an object, which is not part of or generated by the object oriented software application, that provides a connection between the object oriented software application and the runtime engine, and that accesses data from the database in accordance with the map

MicroStrategy advances a definition of “interface object” that is largely consistent with the *Microsoft I* court’s construction, but adds restrictions derived from the patent claims and specifications. *See* 2012 WL 3682915, at *7. DataTern proposes a construction that is consistent with language that the Federal Circuit has used to describe the term. *See DataTern, Inc. v. Epicor*, 599 Fed. Appx. at 950; *Microsoft II*, 755 F.3d at 907.

1. <u>Litigation History</u>

As with other terms, in defining "interface object" the Court does not write on a blank slate. A brief review of the litigation history is warranted. In *Microsoft I*, the New York district court construed the phrase "to create at least one interface object" to mean "to generate code for at least one class and instantiate an object from that class, where the object is not part of or generated by the object oriented application and is used to access the database." *See* 2012 WL 3682915, at *8. Following the New York district court's ruling, DataTern conceded in this case that MicroStrategy's product did not infringe if the term "create" meant "to generate code for at least one class and instantiate an object from that class." Docket No. 39. The Court entered summary judgment of non-infringement based on DataTern's concession. *Id.*

On appeal, the Federal Circuit determined that the Court had misconstrued the term "to create at least one interface object." *DataTern, Inc. v. Epicor,* 599 Fed. Appx. at 951. The Federal Circuit found that the patent did not limit the creation of an interface object to the generation of code, and construed the term to mean "to instantiate at least one interface object from a class." *Id.* However, it did not address the other limitations adopted by the New York district court, including whether an "interface object" "is not part of or generated by the object oriented application and is used to access the database." *See id.* at 950 n.1. In the background section of the court's opinion, the Federal Circuit stated that "'interface objects' . . . act as intermediaries between the object oriented application and the relational database." *Id.* at 950. That language is identical to language the Federal Circuit used to describe interface objects in the background section of *Microsoft II.* 755 F.3d at 907.

2. **“Not Part of or Generated by the Object Oriented Software Application”**

MicroStrategy seeks to limit the scope of “interface objects” only to those that are “not part of or generated by the object oriented software application.[8] The *Microsoft I* court accepted that limitation. That court found that during the patent’s reexamination, the patentee “disclaimed that interface objects were part of or generated by the object oriented application.” *See Microsoft I*, 2012 WL 3682915 at *8. The court went on to examine the prosecution history: “In a supplemental reply submitted in [the reexamination] proceeding, [the patentee] stated: ‘The [prior art] user objects are not to be confused with the interface objects recited in Claim 1 of the present patent, which interface objects are not part of nor generated by the object oriented software application.’ This disclaimer is binding.” *See id.* (citations omitted); *see also* Docket No. 131-21, Patentee Supp. Response, at 10–11.

This Court agrees with that analysis. Accordingly, the term “interface object” will be construed to include the requirement that it is “not part of or generated by the object oriented software application.”

3. **“That Provides a Connection between the Object Oriented Software Application and the Runtime Engine”**

MicroStrategy contends that an “interface object” provides a connection to the runtime engine based on the language of the claims and specifications. That limitation was not at issue in *Microsoft I.*

The description of the invention in the patent states that “interface objects 20 are employed by the object oriented software application 22 to access the relational database 16 via a

[8] DataTern contends that the phrase should be omitted from the definition because “not part of” is confusing and would not assist the trier of fact in understanding the term. *See* Docket No. 198, Pl. Post-Hearing Brief, at 7. However, the patentee used that exact phrase to distinguish prior art, and DataTern itself used the phrase to define “runtime engine” in *Microsoft I*. *See* Note 14, *infra.*

runtime engine 24." '502 patent col. 2 ll. 35–38. On its own, that sentence is somewhat ambiguous. A fair reading of the sentence is that interface objects access the database via the runtime engine. However, it could also be read to mean that the object oriented software application accesses the database via the runtime engine without any intermediaries. In the former case, interface objects provide the connection between the object oriented application and the runtime engine, while in the latter they do not necessarily serve that function.

Although the language is ambiguous, standing alone, the patent provides further guidance. The sentence in question refers to Figure 1, which "illustrates use of the map to generate interface objects that are employed by a runtime engine and an object oriented software application to access a relational database." '502 patent col. 2 ll. 13-16.

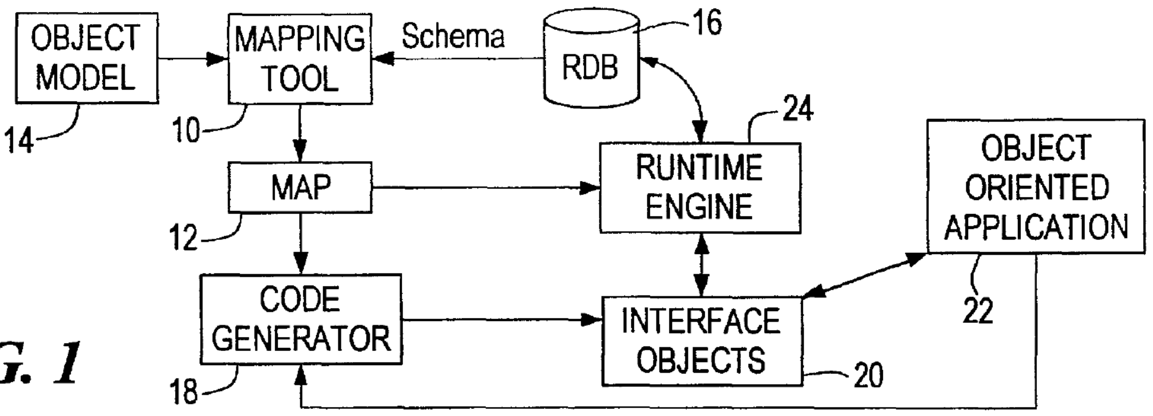


FIG. 1

Figure 1 depicts two ways in which the runtime engine connects to the object oriented application: through interface objects directly—via the pathway from 22 to 20 to 24—or through interface objects indirectly by way of the code generator—via the pathway from 22 to 18 to 20 to 24. In either case, interface objects provide the connection.[9] As MicroStrategy's proposed

[9] Notably, DataTern conceded as much in the *Microsoft I* proceeding, stating that "the invention makes the connection via the runtime engine and the interface objects to pull the data that is needed for the object oriented application from the relational database." Docket No. 131-32, *Microsoft Corporation, et al v. DataTern, Inc.*, Markman Hearing Transcript, July 30, 2012 at 27:19-23.

limitation is supported by the specification, and "most naturally aligns with the patent's description of the invention," it is the correct construction. *Phillips*, 415 F.3d at 1316 (quoting *Renishaw PLC v. Marposs Societa' per Azioni*, 158 F.3d 1243, 1250 (Fed. Cir. 1998)).

4. <u>"That Accesses Data from the Database"</u>

MicroStrategy contends that an interface object "accesses data from the database." [10] Principally, it points to the claim terms in independent claims 1 and 10 as support for that assertion. Claim 1 states that the runtime engine "invokes said at least one interface object with the object oriented application to access data from the relational database," '502 patent col. 8 ll. 1–3, while Claim 10 states that the "runtime engine invokes said at least one interface object to access data from the relational database," *id*. col. 8, ll. 36–37. That language is consistent with the patent description, which (as noted) states that "[t]he interface objects 20 are employed by the object oriented software application 22 to access the relational database." '502 patent col. 2 ll. 35–37. According to the plain terms of the claims, interface objects access data from the database.

DataTern's position actually supports that finding. DataTern points to the same language on which MicroStrategy relies in support of its contention that "interface object" should be construed to mean "act as intermediaries between the object oriented application and the relational database." *See* Pl. Rebuttal Brief at 5–6. Again, DataTern's language mirrors the language the Federal Circuit used to describe interface objects in both *Microsoft II* and on appeal in this case. *Microsoft II*, 755 F.3d at 907; *DataTern, Inc. v. Epicor*, 599 Fed. Appx. at 950. The definition of "interface object" was not before the court in that case. However, in its background

[10] The *Microsoft I* court accepted that an interface object "is used to access the database." It did not consider whether the interface objects "access data" from the database, or whether they access data "in accordance with the map." Although the *Microsoft I* court made a finding concerning that issue, it did not detail its reasons.

section describing the invention, the court stated that interface objects "act as intermediaries between the object-oriented application and the relational database." *Id.* That is doubtless true. More specifically, the way that interface objects act as intermediaries, as DataTern concedes, is that they are "used in the process of accessing data from the relational database." Pl. Rebuttal Brief at 5. That limitation is therefore consistent with the claims themselves and with the Federal Circuit dicta, and the Court will adopt it.

5. "In Accordance with the Map"

MicroStrategy further contends that interface objects access data "in accordance with the map." No such language is contained in the claim terms. Instead the claims recite that the map is "employ[ed]" in order to "create at least one interface object." '502 patent col. 7 ll. 59–60; *id.* col. 8, ll. 32–33. The language stating that the map is "employed" to create interface objects is broader than MicroStrategy's proposed construction that interface objects access data "in accordance with the map." Furthermore, it does not necessarily follow that because the map is used to create interface objects, the objects continue to act in accordance with it. The Court will not read in a limitation that the claim terms do not require.

6. Conclusion

For the foregoing reasons, the term "interface object" will be construed to mean "an object, which is not part of or generated by the object oriented software application, that provides a connection between the object oriented software application and the runtime engine, and that accesses data from the database."

F. Runtime Engine

	Plaintiff's Proposed Construction	**Defendant's Proposed Construction**
"Runtime engine"	software that the object oriented software application uses to access	software that (i) the object oriented software application depends on to

	the relational database	run, (ii) must be running to execute the object oriented software application, (iii) uses the map in its processing, and (iv) is not part of the object oriented software application

MicroStrategy proposes a definition of "runtime engine" that is the same as the one accepted by the *Microsoft I* court. *Microsoft I*, 2012 WL 3682915 at *9. DataTern contends that its definition is consistent with the claim terms and specifications, and that MicroStrategy's definition misreads the prosecution history to improperly import limitations.

The prosecution history supports all four restrictions proposed by MicroStrategy.[11] Among other things, the patentee distinguished *Chang* on the basis that the '502 patent discloses the use of a "runtime engine," while *Chang* discloses the use of a "runtime environment." Docket No. 131–24, Patentee's Response and Proposed Amendments at 15–16. The patentee submitted the affidavit of Mark Eisner in support of that assertion, who described the difference between the two concepts as follows:

> An 'engine' has the meaning of a stand-alone service that enables a specific application or application set to operate. An 'environment' connotes all of the functionality generally available to run a general set of applications . . . The term 'engine' conveys a notion of an encapsulated block of functionality that enables or helps an application to run.

Docket No. 131–25, Eisner Aff. at 3–4. Eisner went on to state that a runtime engine "connotes a service that delivers specific functionality that will enable an application to run," and that, in the context of the patent, "the claimed runtime engine refers to a specific service, or block of functionality, that operates during runtime execution of the object-oriented software application."

[11] The patent itself supports MicroStrategy's proposed restriction (iii) ("uses the map in its processing"). The description of the invention states that the runtime engine "uses the map 12 to drive its processing." '502 patent col. 2 ll. 37–38.

Id. at 4. Based on that affidavit, the patent examiner agreed that the runtime environment of *Chang* was distinct from the runtime engine disclosed in the ’502 patent. Docket No. 131-35, *Ex Parte* Reexamination Communication Transmittal Form at 2. The examiner found further support in a PCMAG.com reference that defined a runtime engine as “[s]oftware that certain applications depend on to run in the computer. The runtime engine must be run . . . to execute.” *Id.* at 8.[12] In commenting on the examiner’s findings, the patentee submitted that it did “not disagree with the Examiner’s indication that . . . the Eisner Affidavit is supported by the PCMAC.com (sic) reference.” Docket No. 131-36, Patentee’s Comments on Reasons for Patentability and/or Confirmation at 1–2.

In an earlier filing, the patentee distinguished *Henninger* for “fail[ing] to teach ‘utilizing a runtime engine’” that is “separated and abstracted from the software application.” Docket No. 131-21, Patentee’s Supplemental Response at 3–5. In support of that assertion, the patentee pointed to an agreed-upon construction of “runtime engine” in prior litigation as “a *dedicated* software routine, which uses the map in its processing, that accesses the relational database during the time period during which the object oriented application is running.” *Id.* at 5.

The *Microsoft I* court considered the evidence described above in evaluating SAP’s proposed definition of “runtime engine,” which is identical to the one advanced by MicroStrategy here. *See Microsoft I*, 2012 WL 3682915 at *9. Based on the prosecution history, the court found that the proper construction was the one proposed by SAP: “[s]oftware that (i) the object oriented software application depends on to run, (ii) must be running to execute the

[12] There are apparently words missing from the PCMAG.com reference provided to the Court. Docket No. 131-35 at 8.

object oriented software application, (iii) uses the map in its processing, and (iv) is not part of the object oriented software application." *Id.* [13]

The prosecution history cited by MicroStrategy, which mirrors and expands upon the prosecution history cited by the court in *Microsoft I*, strongly suggests that this Court should adopt the limitations imposed by MicroStrategy's definition. DataTern contends that MicroStrategy and the *Microsoft I* court misread the prosecution history and that its definition confuses a "runtime engine" with a "runtime environment."[14] However, the limitations imposed on the term come directly from the file history and were used to distinguish prior art.

Accordingly, this court will follow the conclusion of the *Microsoft I* court. The term "runtime engine" will be construed to mean "software that (i) the object oriented software application depends on to run, (ii) must be running to execute the object oriented software application, (iii) uses the map in its processing, and (iv) is not part of the object oriented software application."

G. <u>A Map of at Least Some Relationships Between Schema in the Database and the Selected Object Model</u>

	Plaintiff's Proposed Construction	**Defendant's Proposed Construction**

[13] The *Microsoft I* court concluded as follows: "[The] Eisner affidavit stated that the runtime engine was not the 'runtime environment' referred to in the prior art reference. Eisner also stated that a runtime engine 'connotes a service that delivers specific functionality that will enable an application to run,' and that it 'operates during runtime execution of the object-oriented application.' The examiner agreed with the distinction Eisner put forward and provided a definition of 'runtime-engine" from PCMAG.COM.' . . . This definition states that runtime engine is '[s]oftware that certain applications depend on to run in the computer' and that 'must be running . . . to execute' the software application. The Examiner made this definition part of the file wrapper and therefore intrinsic evidence." *Microsoft I,* 2012 WL 3682915 at *9.

[14] DataTern contends that the limitations imposed by MicroStrategy's definition are "redundant and confusing." Pl. Claim Construction Brief at 21. For example, DataTern complains that MicroStrategy's proposed construction creates a new issue as to how one of skill in the art would determine that something is 'not part of' the object oriented application. However, in *Microsoft I,* DataTern used almost this precise language in its proposed construction. It proposed, "[s]oftware, which is not directly part of the object-oriented application, that the object oriented application uses to access the relational database." *Microsoft I,* 2012 WL 3682915 at *9. DataTern has reprised the latter half of its previously favored definition before this Court, while abandoning the former half and complaining that the language it used is confusing.

"A map of at least some relationships between schema in the database and the selected object model"	no construction needed	a declarative structure that describes at least one association between the objects and attributes of an object model and/or tables, rows, and/or columns in the database classes

MicroStrategy proposes to construe the phrase "a map of at least some relationships between schema in the database and the selected object model" consistently with limitations the patentee used to distinguish prior art. DataTern contends that no additional construction is needed. The parties dispute whether the term "map" in the context of the '502 patent means a "declarative structure." The parties further dispute whether the phrase "between schema in the database and the selected object model" requires further explanation.

1. <u>"Map"</u>

MicroStrategy contends that the only definition consistent with the prosecution history is one that specifies that a "map" is a "declarative structure." DataTern contends that the term "map" needs no further construction and that "MicroStrategy's construction only makes the claim less clear and less precise." Pl. Rebuttal Brief at 23.

The prosecution history supports the language that MicroStrategy now advances. During reexamination, the patentee distinguished a prior art reference (*Bapat*) for failing "to teach the step of 'generating a map.'" Docket No. 131-18, Patentee Response and Proposed Amendment at 19. The examiner stated that although *Bapat* had provided a "mechanism for mapping," the patentee distinguished that mechanism, stating that "*Bapat* provides no discussion or teaching for actually generating a declarative structure that describes as least one association between the objects and attributes in the object model and/or tables, rows, and/or columns in the database, and only generally refers to mapping object instances to tables." *Id*. Similarly, the patentee distinguished *Keller* by stating that "Keller is contrary to the claimed invention, which generates a map, creating a declarative structure that describes at least one association between the object and attributes in the object model and/or tables, rows, and/or columns in the database. Keller fails to teach this step." *Id.* at 17.

Despite this history, DataTern suggests that the phrase "declarative structure" does not provide any additional clarity to the term "map." According to DataTern, even a sentence constitutes a "declarative structure" because it both declares something and uses grammar as its structure. However, that assertion appears to be belied by the prosecution history. *Bapat* also disclosed a mechanism for mapping, but the patentee successfully distinguished the mapping in *Bapat* from the mapping of the '502 patent in part by stating that the '502 patent taught a method for generating a "declarative structure." In addition, the McGoveran declaration suggests—and DataTern does not direct the Court to any contrary evidence—that in the context of the patent a declarative structure "must be realized as some form of metadata or data independent of program code and not merely embedded in or realized as a procedural program." McGoveran Rebuttal Dec. ¶ 14.

The prosecution history is clear and consistent, and controls the resolution of this dispute. "Claims may not be construed one way in order to obtain their allowance and in a different way against accused infringers." *Southwall Techs., Inc. v. Cardinal IG Co.*, 54 F.3d 1570, 1576 (Fed. Cir. 1995). DataTern is bound by the statements the patentee made to distinguish prior art. Accordingly, in the context of the disputed phrase, a "map" is a "declarative structure."

2. **"Between Schema in the Database and the Selected Object Model"**

DataTern contends that the phrase "between schema in the database and the selected object model" requires no additional construction. In its initial briefs, MicroStrategy proposed a construction that substantially mirrored the language used by the patentee during reexamination:

"between the objects and attributes of an object model and/or tables, rows, and/or columns in the database."[15] In its responsive claim construction brief, MicroStrategy states that it would also be amenable to a simpler construction that is substantially the same as the original construction, in which the phrase is construed as "between the attributes of classes of an object model and columns in the database schema." Def. Responsive Brief at 15. As MicroStrategy notes, the parties' difference on this point appears to be one "of form rather than substance." *Id.* at 14.

MicroStrategy's revised proposed definition is consistent with the patent in some respects, but it appears to be overly restrictive. The patent states that "[m]apping the object model to the relational database schema includes mapping a class attribute to a table column, mapping a class attribute to a 1-1, 1-N, or N-N relationship, and mapping class inheritance to rows within a table or across tables." 502 patent col. 2 ll. 44–48. By comparison, MicroStrategy's revised definition restricts this mapping to "between the attributes of classes of an object model and columns in the database schema," which does not encompass the mapping of class inheritance to rows within a table.

Its original definition, "between the objects and attributes of an object model and/or tables, rows, and/or columns in the database," does not so restrict the application of the term. That phrase, although not a model of clear prose, is consistent with the prosecution history as detailed in the previous section. Therefore, the Court will adopt the phrase advanced by the patentee during reexamination.

[15] MicroStrategy's proposed language differs from the patentee's assertion during reexamination only with respect to a preposition. The patentee used the phrase, "between the object and attributes *in* the object model and/or tables, rows, and/or columns in the database," while MicroStrategy uses the word "of" in place of "in." Docket No. 131-18 at 18, 19.

3. **Conclusion**

For the foregoing reasons, the term “a map of at least some relationships between schema in the database and the selected object model” will be construed to mean “a declarative structure that describes at least one association between the objects and attributes of an object model and/or tables, rows, and/or columns in the database.”

IV. **Conclusion**

For the foregoing reasons, the disputed claim terms are construed as follows:

1. the term “object oriented software application” means “software application organized as a collection of classes;”
2. the term “class” means “a definition that specifies attributes and behavior of objects, and from which objects can be instantiated;”
3. the term “object” means “an entity comprising attribute values (i.e., data) and methods (i.e., functionality) that can act on said attribute values;”
4. the term “object model” means “a template with a predetermined standardized structure both relating to an object-oriented software application and including object classes;”
5. the term “interface object” means “an object, which is not part of or generated by the object oriented software application, that provides a connection between the object oriented software application and the runtime engine, and that accesses data from the database;”
6. the term “runtime engine” means “software that (i) the object oriented software application depends on to run, (ii) must be running to execute the object oriented software application, (iii) uses the map in its processing, and (iv) is not part of the

object oriented software application"; and

7. the term "a map of at least some relationships between schema in the database and the selected object model" means "a declarative structure that describes at least one association between the objects and attributes of an object model and/or tables, rows, and/or columns in the database."

So Ordered.

/s/ F. Dennis Saylor
F. Dennis Saylor IV
United States District Judge

Dated: February 7, 2017